Everybody has heard about the cool new things that can be done using Machine Learning, and I’m here to talk to you about how this can be achieved using Vulkan.

1. Hi everyone – I am Pierre Boudier- I work on graphics drivers at NVIDIA and I am the chair of the Vulkan Machine Learning Subgroup at Khronos. This group was formed in spring 2019 and was tasked with bringing a machine learning solution to Vulkan.

2 The Machine Learning subgroup received a number of design contributions from hardware vendors, and examined requirements from both software and hardware vendors. The design goal was to enable the eco system with an easy integration with the rest of the Vulkan API functionality. Machine Learning is still an active area of research, and new neural network architectures are coming out every year. While there are a number of natural building blocks like matrix multiplication, convolution, relu/sigmoid/tanh activations, there is a combinatorial explosion of architectures, as well as new arithmetic concepts being added every week. For instance locality sensitive hashing, sparse matrices, quantized weights, binary weights, ...

Software vendors were also clear—there is a need for a solution in the Vulkan eco system which has lower overhead than using interop with some other API, and without having to install large external frameworks (like python).

Luckily for us, we did not have to start from scratch as Vulkan already had a compute API capability built-in from day one. Additionally, some of the new functionality specifically targeting ML had some cross over requirements with traditional graphics/compute usage; as a result, new extensions were introduced since the initial release of Vulkan that benefit directly Machine Learning specific workflows.

3. So, what does it mean to do ML in Vulkan?
In a typical workflow, a data scientist will create a neural network by training it on a desktop or in the cloud, using high end accelerator systems. Those networks are then exported using some exchange format, which include both the definition of the computational graph, and the values of the parameters. Two of those formats are NNEF and ONNX; NNEF being defined and supported by Khronos.
At runtime, the application would load those formats and generate some compiled code that can be executed in a step called “inferencing”. The application would use the Vulkan API via compute shaders to offload the computation on the GPU accelerators.

4. So how does one use compute shader and what precisions should be used?
Machine learning is about running a computational graph, which by default is usually done in 32 bits in the python frameworks. The precision has to match between training and inferencing. There is actually very little difference between training and inferencing from an execution point of view: training is just running two extra graphs to backward propagate the gradients and to update the parameters.
For many architectures, it is actually possible to use lower precision than the full 32bit, which results in both faster execution, and reduced data storage. 16 bit floating can be used, by monitoring the magnitude of the gradient during training to avoid round to zero for small values. 8 bit integer can also be used in quantized network, which are generated via a distillation phase. Vulkan supports those data types natively since the version 1.2, or via the extensions VK_KHR_shader_float16_int8 and VK_KHR_8bit/16bit_storage.

An upcoming extension SPV_KHR_integer_dot_product allows to write optimized kernels using integer math without having to rely on compiler optimization.

5. Next, we looked into ways to improve Vulkan compute for ML
The first upcoming extension, VK_KHR_workgroup_memory_explicit_layout improved the efficiency of loading data into shader memory without blocking the arithmetic units.

The second upcoming extension, VK_EXT_ML_primitices, exposes common building blocks that are found in the layer of the popular neural nets, which the driver can optimize more thoroughly.

More details will be shared for those two extensions once they become publicly available.

A third extension is already available, VK_NV_cooperative_matrix, and can be used in a shader to directly access dedicated hardware optimized for matrix multiplications. This is typically done for matmul and convolution layers using 16bit floating point precision.

6. How does one get started?

An easy way to run neural networks in Vulkan is to compile the computational graph thanks to a generic compiler; we will mention two open source projects:
- TVM: originally started at the university of Washington, will compile networks into Spir-V that can be run on top of the Vulkan API
- MLIR/IREE: is a multi level IR (MLIR) infrastructure developed inside LLVM, associated with an IR execution engine (IREE), which can generate both the spir-V code and the associated Vulkan API code.

7. Looking into more details of MLIR/IREE
Google has been collaborating closely at Khronos when developing their Vulkan based inferencing solution.
The compiler starts from a generic XLA-HLO description of the network, which can be generated from the tensorflow/pytorch python frameworks; it will then lower successively into multiple levels of abstraction, each one of them targeting a domain specific dialect. The last output is a combination of the spir-v code that can be compiled to a specific GPU ISA by the driver, and a list of Vulkan API calls in order to submit those kernels and manage the associated allocations and synchronization points.
The compiler has a unified view of the entire computational graphs, and can optimize efficiently across layers, and schedule dispatching of operations without unnecessary synchronizations.

8. Other third-party frameworks do provide acceleration using Vulkan
Let's have a look at a few:
Alibaba provides the MNN library that can do both inferencing and training on mobile platforms. It is being used in many of the most popular Alibaba applications like Tобao, Tmall, Youku, DingTalk, etc
Tencent provides the NCNN library that can do inferencing on multiple platforms, mobile and desktop. It is being used by many of the popular Tencent applications such as WeChat, Pitu, QQ, Qzone etc
Ax inc provides a SDK called Ailia which has highly optimized implementation of many of the popular networks across multiple platforms, and provides multiple support options for companies interested in deploying neural nets.

9. There are options to do machine learning in Vulkan outside of purely running Neural Nets.

 VkFFT is an open source project started at Julich which provides highly efficient fast fourier transform kernels, with lots of possible parametrizations. It has shown great results for image resampling, image registration, and even in generic signal processing for physics simulation

 Kompute is a framework that allows writing kernels and algorithms for deployment on multiple vendor graphics card, with integration in both c++ and python.

 Unity provides ml-agents which can be used to train intelligent agents within the Unity game engine based on reinforcement learning (proximal policy optimization PPO, and soft actor critic SAC).

10. So, Here is our Call to Action
This was an overview of the Vulkan Machine Learning capabilities that are now available: production drivers have been shipping for a while on many devices.

We are welcoming feedback from application or framework developers; if you have needs which are not addressed at all by the existing solutions, or you have pain points which could be alleviated, do not hesitate to contact us.
We are interested in both neural nets workflows, and other classical machine learning algorithms like random forest, logistic regression, t-sne, k-nearest neighbor …
My email address is pboudier@nvidia.com.

You could decide to join the Vulkan advisory panel, or even become a Khronos members.